AR-008-799

DSTO-TR-0146

# Functions For Processing Synthetic Aperture Radar Imagery

Robert Whatmough and
Paul Whitbread

DEPARTMENT OF DEFENCE
DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

# Functions For Processing Synthetic Aperture Radar Imagery

*Robert Whatmough and Paul Whitbread*

**Information Technology Division**
**Electronics and Surveillance Research Laboratory**

DSTO-TR-0146

**ABSTRACT**

**Technical Report**

The INGARA synthetic aperture radar system can potentially benefit from higher level algorithms and software to enhance its basic imagery so that important features are obvious to human operators. These could include filters for removing speckle, zooming functions for correcting the aspect ratio of the imagery and classifiers for distinguishing ground cover types. The system could also make use of a development environment in which such functions can be easily developed and trialled. This Note describes the Khoros image processing system and functions added to that system by researchers in ITD that could be of use in processing INGARA data. It also includes results of tests of the effectiveness of these functions in enhancing sample INGARA data.

**APPROVED FOR PUBLIC RELEASE**

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

*Published by*

*DSTO Electronics and Surveillance Research Laboratory*
*PO Box 1500*
*Salisbury South Australia 5108*

*Telephone: (08) 259 7053 division phone number*
*Fax: (08) 259 5619 division fax number*
*© Commonwealth of Australia 1995*
*AR-008-799*
*January 1995 approval to publish date*

## CONTENTS

## FIGURES

## EXECUTIVE SUMMARY

The INGARA synthetic aperture radar system developed by Microwave Radar Division is now producing images. These images will need to be further processed to make important detail visible to human operators. This report has been written to describe some of the image processing software and systems presently available to carry out such processing: it details appropriate algorithms and software currently held by Information Technology Division, and also presents an effective development environment for creating and testing new algorithms.

The Khoros image processing system has been developed at the University of New Mexico and is freely distributed over computer networks. It provides a wide selection of basic image processing functions and a graphical user interface through which these functions can be combined to build more complex operations. Users can develop their own specialist functions and add them to those provided.

Standard Khoros functions relevant to INGARA processing perform image display, several forms of filtering, grey-level manipulation, geometric rescaling, and the computation of local statistics for texture analysis. Information Management Group researchers have added functions that perform new forms of these operations, and also automatic image thresholding, supervised texture classification by artificial neural network and cleaning up errors in classifier output.

The above functions were tested on INGARA images, for their ability to alter the aspect ratios of images, to remove radar "speckle" without destroying other features of possible importance, to detect bright objects as "targets" of interest, and to classify surface regions using texture measurements. Various combinations and sequences of grey-level transformation, rescaling and filtering were tried. The output was inspected visually, and quantitative measures of speckle reduction and point target blurring were computed where relevant.

Not all functions were suitable for the data used. Various functions and combinations are recommended for geometric rescaling, speckle reduction, classification and target detection.

# 1. INTRODUCTION

The INGARA (formerly AuSAR) synthetic aperture radar (SAR) system developed by Microwave Radar Division (MRD) (Stacy & Burgess, 1994) is now producing images. These images may contain hundreds of megabytes of data and will need to be further processed to make important detail visible to human operators. This report has been written to describe some of the image processing software and systems presently available to carry out such processing: it details appropriate algorithms and software currently held by Information Technology Division (ITD), and also presents an effective development environment for creating and testing new algorithms.

The first processing step that must be applied is to transform the received radar signals to the spatial domain. It is assumed here that this step and any basic enhancements, such as "multi-look" processing to reduce radar speckle, have already been adequately performed. Further processing that can then be usefully applied includes speckle removal, geometric manipulation such as aspect ratio correction and zooming, and classification of ground cover types from the statistical variation of image grey level ("texture"). The report provides brief descriptions of a number of algorithms developed by the authors and others for each of these steps, and describes their performance on sample images supplied by MRD in various combinations and with various parameter settings. Based on the results of these experiments, recommendations are made of suitable algorithms and appropriate parameter settings for processing of similar data in future.

The algorithms described here have all been implemented within the Khoros image processing system, and have been supplied to MRD along with Khoros. As Khoros provides an excellent environment for developing image processing software, the report also briefly describes this system and indicates how new algorithms can be added to it. It is recommended that future development of algorithms for Ingara be done within this or an equivalent comprehensive development system for image processing.

Finally, while the algorithms presented here have been selected as being appropriate for processing SAR data, the authors make no claim that the report is a comprehensive survey of all available techniques. Users are referred to the literature for more extensive coverage (see, eg, the range of algorithms proposed for speckle removal reviewed in McDonald, 1988).

# 2. THE KHOROS IMAGE PROCESSING SYSTEM

The Khoros system is the result of an ongoing project at the University of New Mexico to develop a sophisticated and comprehensive software environment for image processing (eg

Rasure et al 1990, Konstantinides & Rasure 1994). The version described here is Khoros 1.5, but a major update with many new features, Khoros 2.0, is due for general release very shortly.

While the Khoros copyright is now under control of Khoral Research Inc., the software is freely distributed through FTP servers worldwide; this includes all source code and extensive documentation. The copyright prohibits redistribution of Khoros by users (eg embedded in a larger software system), but users freely interchange their own software extensions with others who have acquired the standard system.

Electronic mail and the Usenet computer news network (see the newsgroup comp.soft-sys.khoros) allow users to raise problems in the use of Khoros with the writers and each other. The writers give free advice, and other users announce development of and provide access to their own extensions, through these mechanisms.

Khoros runs on a variety of machines with the UNIX operating system and X-Windows software. It includes functions for processing 1-D (signal) and 2-D (image) data, stored in a format ("VIFF") that allows bit, integer and floating-point data (including complex data for transforms) and the inclusion of colour maps (or look-up tables).

Finally Khoros 1.5 handles all images in the same manner (eg it reads the entire image into memory), and therefore can have problems with very large images such as those produced by SAR and satellite imaging systems. Khoros 2.0, however, has more sophisticated data format and management facilities to specifically handle large images.

## 2.1 Basic Khoros functions

The functions supplied with Khoros are very numerous and cover a wide range of applications. Categories of functions relevant here include:

Conversion between VIFF and other formats, including ASCII text
Image display or conversion to PostScript files
Data type conversion
Applying lookup tables to data
Applying median or convolution filters
Speckle removal (one algorithm)
Discrete Fourier transform and its inverse
Histogram manipulation for contrast adjustment
Geometric manipulation and warping
Subimage extraction
Texture analysis (several algorithms)

Classification and segmentation

Not all the "standard" functions are ideal for the present application; some have been modified here to make them more useful.

Each function can be invoked by a standard UNIX-style command, which would typically specify one or more input images, an output image and one or more parameter values. Shell scripts can be used to combine functions, but for experimentation it is usually more convenient to use Cantata.

## 2.2 Cantata, a graphical interface

Cantata is a program, supplied as part of Khoros, that allows functions to be combined into a "workspace" and viewed together as a flowchart. Single-function steps can be added, deleted or modified, connections between them (which represent intermediate images) can be re-arranged, and the whole workspace or individual steps executed as required. Temporary disk files are created automatically to hold intermediate images.

To make this possible, each function must have defined for it a menu or "pane", in which are entered parameter values and names of image files (when they are not temporary files to be defined by the connections in the flowchart). The functions must also be entered into a file structure called a "toolbox" so that they may be located by short descriptions in a hierarchical menu search.

For example, selecting the toolbox "IMAGE PROCESSING" produces a menu of categories. Selecting the category "Subregion" produces a window which includes a menu of functions. Selecting the function "Extract Subimage" sets up the window to receive parameters and image names for the *vextract* function. This function copies part of an image to a new image, and its parameters identify where and how big that part is. When values are entered, the window can be converted to a "glyph" or flow-chart symbol and placed on the chart. Its input and output connectors can then be joined to other glyphs as required. The parameter values can be recalled and modified later.

At the end of a work session, a workspace can be saved, together with all the temporary image files in it (in compressed form). When Cantata is next run, a saved workspace can be restored, and any intermediate data that was saved with it recovered intact so that later steps of the workspace can be modified and re-run without re-running earlier steps.

## 2.3 User extensions

Khoros includes software that helps a more sophisticated user to add functions to Khoros, and encourages appropriate styles and documentation. There are several approaches to doing this, but among the simplest is to supply three files:

1. A ".pane" file which defines the appearance of the pane where parameter values would be entered if the function were used through Cantata, the types and numerical bounds for parameters, names and text descriptions of parameters and the name of the function to execute;

2. A ".prog" file which contains, in a standard format, code fragments (in standard C) for reading the input images, checking parameters and images against one another for consistency, processing the data and writing the output images. Documentation (manual pages and help data) also appears here;

3. A ".conf" file which helps define the file structures for the function, its code and documentation, and the toolbox to which it belongs.

The ".pane" and ".prog" files are the minimum information necessary to define a new function to a user at another installation.

A utility program *preview* checks the ".pane" file and allows the developer to experiment with a dummy version of the parameter window.

Another utility *ghostwriter* processes the ".prog" and ".pane" files to produce the necessary source code files and the makefile for compilation and linking. Executing the makefile produces a function executable by command.

A third utility *kinstall* makes the function available as an item in a toolbox through Cantata. It leads the developer through various editing steps to set up compatible menu entries in the right order.

Functions that perform image display, and perhaps user interaction with the display, are possible but harder to install. Less utility support is available for these.

Further details of the above are available in the manuals supplied with the Khoros system.

# 3. RELEVANT STANDARD KHOROS FUNCTIONS

The following functions are supplied with Khoros and are relevant to INGARA processing, although some have been superseded by improved locally-written functions (see next section). Basic conversion and arithmetic functions will not be described here.

## 3.1 Image display (*editimage* and *put_update*)

*editimage* and *put_update* both display images on the terminal. If an image is too large to show on the screen, part of the image is displayed at full scale and the whole image is shown at reduced scale in a second window ("pan window"). An overlayed box in the pan window can be moved around to select the area that is shown at full scale.

Both functions (as executed from Cantata) regularly check their input files for new versions. Once started, they can be left running and will update their displays whenever the functions that supply their input are re-executed.

*editimage* allows manipulation of its display. It can show numerical pixel values, change the lookup tables, and enlarge part of what it displays by replication of each pixel (to make an *nxn* block of identical pixels). It can also produce an annotated version of an image.

## 3.2 Median filtering (*vhmed*)

*vhmed* applies a median filter to an image. The value of a pixel in the output is the median of the values in a rectangular region of the input, of odd dimensions, centred on the same pixel. The algorithm used takes full advantage of overlap between successive positions of the rectangular region.

Median filtering can reduce noise in an image without the blurring of real features that convolution (FIR) filters produce.

## 3.3 Speckle removal (*vspeckle*)

*vspeckle* applies the "geometric" speckle-removal filter of Crimmins (1986) to an image. This filter (in effect) treats an image as a three-dimensional surface and iteratively smooths cross-sections through the surface.

Successive iterations lower the high regions, and raise the low regions, at rates that decrease with region size. After a few iterations, then, small weak features are removed but strong or broad features are not yet greatly affected. The number of iterations is a parameter and should be chosen so that the filter stops just as the speckle disappears.

Use of this algorithm assumes that the speckle intensity is the same everywhere. This may never be exactly true but may be roughly true when the image grey-level represents log(power of radar return).

## 3.4 Convolution filtering (*vconvolve*)

*vconvolve* applies a convolution (or FIR) filter to an image. The value of a pixel in the output is a weighted sum (in practice, usually a weighted mean) of the values in a rectangular region of the input, with its bottom right corner at the same pixel.

The size of the region and the weights to be used are specified as a second input image, the "kernel". Input values that are required but that lie outside the range of the input image are assumed to be zero. (This gives the result dark margins at the top and left.)

As in signal processing, convolution filters can be designed to emphasise or weaken certain frequencies, find derivatives and so on. They are not well suited for the high noise levels of SAR speckle.

The implementation of this function makes it inconvenient to use when what is really wanted is a symmetric combination of values on all sides of a pixel. Only a shifted version of that result is available. For an alternative see section 4.2 below.

## 3.5 Mapping data through a map (*vmapdata*)

The *editimage* and *put_update* display functions automatically apply any map (lookup table) in an image to the pixel values for display purposes. Most other functions ignore maps and pass them through unchanged to their output images. *vmapdata* applies the map in an image to its pixel values and produces an output image with new values and no map.

## 3.6 Changing the data type of an image (*vconvert*)

*vconvert* changes the current data type of an image to another specified type. The values can be optionally scaled; if the output is 8-bit integer data, values can be scaled and offset to the range 0-255 automatically.

## 3.7 Rescaling an image (*vexpand, vresize, vwarp*)

*vexpand* makes a larger version of an image by replacing each pixel by an *nxn* block of identical pixels. (This is the same replication operation as that performed in the zoom window of *editimage*.) For large values of $n$ the result has a distracting "blocky" appearance.

*vresize* makes a larger-scale or smaller-scale version of an input image using values found from the image by bilinear interpolation. The enlargement can be different for the horizontal and vertical directions. The output image has the same size as the input, and parameters specify the coordinates of the pixel that remains fixed during the scale change. A bug in this function presently forces it to work with the centre part of the input image. The result has a smoother appearance than that from *vexpand*.

*vwarp* applies a general polynomial coordinate transformation to an image. It computes the output location of each input image pixel as a polynomial function of its input coordinates, using coefficients taken from two small auxiliary input images (one for each output coordinate). The output values are generated by an area-filling algorithm, which approximates bilinear interpolation (but not very well). The need to set up the two coefficient images makes this function inconvenient to use (though versatile).

## 3.8 Fractal dimension estimation (*vfractal*)

*vfractal* interprets a small region of the image as a fractal function, ie, one whose fluctuations occur in a manner that depends on scale and a parameter termed the "fractal dimension". By computing the ranges of grey levels over areas of different sizes within the region, it estimates this dimension, usually a real number between 1 and 3. It repeats the calculation for overlapping regions to find how the dimension varies across the image. Fractal dimension is one of many measures of image texture.

## 3.9 Computing local statistics (*vspatial*)

*vspatial* computes grey-level statistics in $n$x$n$ windows centred on all pixels in an 8-bit image, forming an output image with one band for each statistic. The windows can have any odd dimension, and the statistics can be any selection from mean, variance, "contrast", "angular second moment", "entropy" and "dispersion". These statistics are intended as measures of texture.

## 3.10 Laws' texture measures (*vtexture*)

*vtexture* implements the kernel processing stage of the Laws texture measures (Laws, 1980). This function convolves the image with a predetermined set of 5x5 masks. To get a texture measure as described by Laws, it is necessary to find the standard deviation of the results of *vtexture* by using the variance operation of *vspatial*, with larger window than the Laws' kernels, eg 11x11. A set of standard deviations from different masks can be used as a vector of attributes from each pixel and classified in the same way as a multi-spectral image.

Under Laws' approach a critical step in deriving texture measures is the selection of appropriate masks. Masks are based on 1D, 5-point kernels which emphasise different frequencies and are called "level" (L5), "edge" (E5), "spot" (S5), "wave" (W5) and "ripple" (R5). Each outer product of two 1D kernels gives a 5x5 kernel with properties allegedly related to those of the 1D kernels. In practice, the spatial features selected by the 5x5 kernels are not so well defined and kernels must be chosen by trial and error to detect a particular texture type.

# 4. FUNCTIONS ADDED BY ITD

The following functions have been written by researchers in Information Management Group in ITD and are available for use in Cantata through various toolboxes.

## 4.1 Quantile filtering (*vquantile*)

*vquantile* applies a quantile filter, a generalisation of the median filter (section 3.2). Of the $n$ values in the rectangular region, it chooses the $k$th smallest. $k$ is the nearest integer to $p(n\text{-}1)+1$, where $0 \leq p \leq 1$ is a parameter of the function. The minimum ($p$=0), median ($p$=0.5) and maximum ($p$=1) are special cases.

Quantile filtering can be considered when median filtering removes strong local returns from a SAR image that might indicate small but strong reflectors.

## 4.2 Centred convolution (*vconvolvc*)

*vconvolvc* applies a convolution filter, and is an replacement for *vconvolve* (section 3.4). The value of a pixel in the output is a weighted sum (usually the weighted mean) of the values in a rectangular region of the input, of odd dimensions, centred on the same pixel.

As for *vconvolve*, the weights are specified in a "kernel" image. When these weights are symmetric, the input image features are not shifted. When input values that lie outside the input image are required, the input image can be treated as though it were extended across its boundaries by reflection there. (Other options are available, but this one best avoids abrupt changes near the edges when the results are for display.)

## 4.3 Abseil filtering (*vabseil*)

*vabseil* (Whatmough, 1990) was written as a variation on the Crimmins filter (section 3.3). It assumes that the input image data is of the form

(true value) x (multiplicative noise) + (additive noise)

where the multiplicative noise has mean 1, the additive noise has mean 0, and both have known standard deviations (given as parameters). For each pair of adjacent pixels (horizontally, vertically or diagonally) it estimates the local grey level from the two pixel values, and hence the local noise level. It then diffuses the pixel values into each other by an amount that depends on their difference relative to the noise level. This step is repeated any desired number of times.

This method was developed using data from the STAR (Sea-ice and Terrain Assessment Radar) SAR system, which featured multi-look processing. It worked fairly well for that data. The multiplicative noise model approximates the real characteristics of SAR speckle.

## 4.4 Intra-cluster average filtering (*vicafil*)

*vicafil* (Whatmough, 1990) was written as an enhancement of the sigma filter (Lee, 1983). Like *vabseil*, it was developed on STAR data. It tries to interpret the image as "patches" of constant value perturbed by known levels of multiplicative and additive noise (specified as parameters).

For each possible patch value $v$ near an input pixel, it computes (through tables constructed in advance) the noise standard deviation $\sigma(v)$, and sets limits $v \pm 2\sigma(v)$. It considers all the values of $v$ for which the input grey level falls between these limits, and finds the value for which the largest number of neighbours of the pixel also have grey levels between the limits. ("Neighbours" of a pixel are all the pixels within a radius specified as another parameter.) The mean of the neighbouring grey levels between the limits is an estimate of the patch value and is made the output value of the pixel.

This filter has a useful side-effect: when noise levels are low, it tends to sharpen blurred edges. It can be applied twice with different parameter settings to reduce noise first and then sharpen.

## 4.5 Contour-smoothing filter (*vcntsmth2, vcntsmth3*)

*vcntsmth2* and *vcntsmth3* are approximate implementations of the non-linear diffusion filter of Alvarez et al (1992). The two different filters use different discretisations of the continuous partial differential equation underlying the non-linear diffusion filter. The filters modify the values of the image so that each segment of a contour of image value moves towards its centre of curvature a distance proportional to its curvature. Small features disappear, and contours become smoother, but broad features are preserved. The process is repeated several times, the count being a function parameter.

The process is not affected by the values of contours, so no preference is given to preserving strong point features that might be important in SAR imagery, but strong edges do tend to be preserved.

## 4.6 Polynomial cleaning filter (*vcleanpol*)

*vcleanpol* is an experimental filter that attempts to distinguish parts of an image with real features from those where the only variation is produced by uncorrelated noise. It expresses the values in a 3x3 region as a combination of orthogonal polynomials, assumes that the higher-order terms are purely noise, and hence estimates the signal and noise powers in the region.

The output value at the centre of the region is then set to

{ (centre input value)x(signal power) + (region mean)x(noise power) } / (total power).

The process is repeated several times, the count being a function parameter.

## 4.7 Setting lookup table from ASCII file (*vsetmap*)

*vsetmap* copies the pixel values of an input image unchanged to an output image, but creates the map (lookup table) for the output image from values in an ASCII file. Optionally it can pass the values of an existing map through the new map instead of just replacing them.

The ASCII file can be created by a text editor or a simple, user-written program. When combined with *vmapdata* (section 3.5), *vsetmap* provides a way of applying any functional transformation to an 8-bit image, pixel by pixel. A colour table can be entered to create a false-colour display of grey level data or of class labels from a classifier.

## 4.8 Smooth zooming (*vsmzoom*)

*vsmzoom* enlarges an input image by integral (but not necessarily equal), parameter-specified factors *m* horizontally and *n* vertically. The output values are chosen so that they vary as smoothly as possible, under the constraint that if the output image is divided into *mxn* blocks, the block averages (or, optionally, the values at the block centres) match the input pixel values. This scheme was proposed by Newsam (1993).

By default, smoothness is maximised by minimising the mean square output of a digital Laplacian operator applied to the image. The operator can be changed under control of a parameter to alter the level ("order") of smoothing. Only one level of smoothing less than the default is allowed, and this gives results more like those from *vexpand* (section 3.7). Higher levels of smoothing produce more "ringing" oscillations near sharp edges and peaks.

The minimisation is done in the Fourier domain, and the input image is assumed to be periodic. This has minor effects at the edges.

This method could be further modified by changing the range or weighting of the local mean of output values that matches an input value in the constraint. The choice of mean or centre value is equivalent to an assumption about the nature of the sampling that produced the original image. The sampling depends on the resolution of the sensor. Edge effects could also be reduced by using the cosine transform in place of the Fourier transform.

## 4.9 Extracting and zooming region of interest (*vextzoom*)

*vextzoom* extracts a specified part of an image and zooms it to any specified size using any of three interpolation methods: nearest-neighbour, bilinear, or "cubic convolution" (bicubic). It is an alternative to *vextract* followed by *vsmzoom*, to *vresize*, and to *vwarp*, and will sometimes be more convenient than any of these. The zoom factors involved may be non-integral and unequal.

The bilinear interpolation method matches that used by *vresize*, and approximately that used by *vwarp*. The cubic convolution interpolation method gives similar results to the centre-value option in *vsmzoom*, but is generally faster.

The interpolation can make use of input values of pixels just outside the region of interest. If these pixels are also outside the image, their values are generated by assuming reflection of values across the image boundaries. No periodicity is assumed.

## 4.10 Thresholding (*vthresel, vhistmap*)

Thresholding classifies a single-band image by dividing the range of grey levels into contiguous sub-ranges, one for each class. Thresholding is not likely to be useful for general classification of SAR images, but may help with the detection of man-made objects if they are much brighter than their surroundings.

Ideally, the grey-level histogram of an image with differently-coloured objects in it will have two or more distinct peaks, each corresponding to one of the objects. These peaks will be separated by valleys, and if the sub-range boundaries are placed within the valleys, thresholding will classify pixels according to which object they belong.

In practice, the grey-level ranges of different objects will overlap, and may be separated only by shallow valleys or "shoulders" on slopes. Because of the overlaps, classification of pixels will be subject to error, and the number of objects will be harder to decide.

*vthresel* chooses the sub-ranges for thresholding, and sets the image map to do the classification. It constructs the histogram of its input image and finds its exponential hull - the closest-fitting function whose value is no less than the histogram value and whose logarithm has only zero or negative second differences. The exponential hull residual (the area between the histogram and the hull) is a sensitive indicator of whether the grey level is a mixture of simpler distributions (Whatmough, 1991). The fit is repeated on different parts of the histogram. The sub-ranges for thresholding are then chosen to reduce the sum of their residuals to an acceptable value with as few sub-ranges as possible.

In some images, the grey levels have already been manipulated by earlier processing and there are many spurious features in the histograms that do not indicate suitable sub-ranges for classification. To handle these cases, *vthresel* also allows the set of contour lengths at different grey levels to be computed and used instead of the histogram. The resulting classes will tend to be separated at levels where the contours follow strong edges without detours around noise features.

There are several ways to specify to *vthresel* what is an "acceptable" value for the sum of residuals. The simplest is to set a fraction of the area under the histogram. The desired number of classes (or sub-ranges) can be set, but it is only a maximum - if the grey levels have a unimodal distribution, *vthresel* may chose a smaller number, even just one. The actual number chosen can be assigned to a Khoros "variable" for use as a parameter in later processing.

The output image has the same image data as the input, and a map that assigns each grey level to a class. The classified image can be produced by mapping the image data through the map, using *vmapdata*. It may be displayed by using *vsetmap* to map the thresholding map through another map that assigns a colour to each class 0, 1, 2...

*vhistmap* was written mainly to allow the thresholds (sub-range boundaries) to be examined and compared with the image histogram. Given a single-band image, it produces an image that shows the histogram in black and white, and the colours in the map in a strip along the grey-level axis of the histogram. For an image processed by *vthresel*, then *vsetmap*, this will show the histogram of the original, and the subranges marked with the colours that are used to display the classified (thresholded) image.

## 4.11 Texture (*vharalick*)

*vharalick* calculates nine of the most commonly used texture measures due to Haralick (1973). This function takes a single band grey level image and produces a multiband image with nine values corresponding to each pixel.

The nine measures of texture are functions of a pixel's neighbourhood co-occurrence matrix (see next para.), which is also called the spatial grey level dependence matrix. The texture measures have evocative names chosen to draw a parallel between 1D and 2D statistics; examples are entropy, angular second moment and contrast. (These are not the same as the similarly-named statistics in *vspatial*, section 3.9).

The co-occurrence matrix is a two-dimensional histogram containing entries $P(i,j)$ which are estimates of the probability of finding two pixels with grey levels $i$ and $j$ separated by a displacement $(x,y)$. In *vharalick*, $x$ and $y$ are parameters specified by the user. The estimate of probability is made by counting the occurrences of pixel pairs over a neighbourhood of a size specified by parameters.

Calculation of the texture measures using Haralick's formulation is computationally intensive. For a quick inspection of texture, *vharalick* provides an optionally selectable approximate method of calculating the texture measures using sum and difference histograms (Unser, 1986). The approximations are of variable quality and the approximation for "energy" is of doubtful value. However, the increase in speed of calculation is marked.

The speed of calculation is also related to the number of grey levels in the input image. To increase it, there is an option to specify fewer grey levels: this is intended to be used after first renormalising the input image using *vconvert* (section 3.6).

## 4.12 Supervised classification by neural network (*vptron*)

*vptron* simulates an artificial neural network designed to perform supervised classification; i.e/ given examples of the land cover/target types, each pixel in the whole image is then labelled with the label of the most similar pixel in the examples. *vptron* is a flexible implementation of a multi-layer perceptron (MLP) that has previously been used on multi-spectral data (Whitbread, 1992). The texture features of this implementation are also useful for single band data such as INGARA data.

The simulated MLP is a multi-layer feedforward network with 3 (or fewer) layers of identical nodes. Each node computes a nonlinear function of the weighted sum of its inputs. The MLP learns the distinguishing features of classes by adjusting the weights in these sums in an iterative process. This simulation uses least squares, back propagation and gradient descent to optimise the weights, and is subject to the failings typical of this process (slow convergence, trapping in local minima, etc). Nevertheless it has been found to be a useful classification tool where other methods have difficulty.

*vptron* classifies the image on a pixel-by-pixel basis, but also uses pixels in the neighbourhood of the pixel to be classified to estimate local texture characteristics. The neighbourhood size is adjustable by the user. The user also has controls over the architecture of the simulated MLP, and a number of parameters relating to learning rate. (Sensible defaults for all these parameters are supplied.)

When used as a texture classifier, *vptron* requires three inputs; an image to be classified, an image containing some pixels of known classification (the training image) and an image of labels (the training labels). The image of labels contains a class identifier (1...16) at locations corresponding to known pixels in the training image and zero everywhere else. In most cases, the training image and the image to be classified will be the same.

The prime output of *vptron* is a class map of labels. Optionally *vptron* can also provide a certainty map, giving the probability that each pixel is in a particular class. The trained weights can also be stored for reuse.

## 4.13 Majority filtering (*vmajor*)

*vmajor* applies majority filtering to an 8-bit image. The output value of a pixel is set to the most frequent input value in a window centred on that pixel. This process is most likely to be useful for cleaning up class label images produced by a classifier that mis-classifies some pixels.

Zero values are allowed in the input and are interpreted to mean "unclassified"; they may be produced in the output when no class has a clear majority. The window is approximately circular and its radius can be varied. Other optional parameters can set a "minimum majority" level, allow arbitrary breaking of ties, and allow unclassified pixels to acquire labels from other nearby pixels.

# 5 TESTS ON INGARA DATA

MRD supplied samples of real airborne INGARA data recorded over Cultana Station near Whyalla, S.A. These were 8-bit images of log(power), and, in order to preserve the higher along-track resolution, were produced by single-look processing instead of the usual 4-look processing. The pixel spacing on the ground was therefore about four times smaller in the along-track direction (in this case, horizontally in the image) than in the cross-track direction: consequently displays of the raw images showed strong geometric distortion.

The ratio of powers in adjacent grey levels in these images was estimated at 0.176 dB (N. Stacy, pers. comm.). For single-look data, the standard deviation of power produced by speckle in a featureless area is equal to the mean power, and measurements made in apparently featureless areas on the image gave values of $\sigma/\mu$ between 1.05 and 1.1 .

The functions described in sections 3 and 4 were tested for their ability to:

(i) correct the geometric distortion by stretching the image in the vertical direction without introducing a "blocky" appearance or other artefacts in the result;

(ii) remove speckle without destroying other features, eg point reflectors like buildings, linear features like tracks, and edges of areas of differing cross-section such as water or topographic shadow;

(iii) distinguish surface regions using local properties. (This must be done without removing speckle, for the speckle is assumed to contain more information than just multiplicative noise.)

The quality of the output from (i) was judged by visual inspection only. Measurement of the quality of the output from (ii) is discussed below.

## 5.1 Alternatives for geometric processing

Initially, speckle reduction tests were performed on parts of the raw data and the results were viewed by using *vsmzoom* to stretch them four times in the vertical direction. It was then observed that real features visible to the eye in the raw imagery were generally elongated in the horizontal direction. This suggested that, provided that speckle was first reduced by filtering in the horizontal direction, geometric correction could be achieved by simply extracting every fourth column of the image, without loss of real features, before the main speckle-reduction step.

Tests of speckle reduction were therefore tried on horizontally-reduced sub-images. Mean and median filters (5x1) were tried for the horizontal filtering step, and the output was processed by *vsmzoom* to stretch it both ways for comparison with sub-images processed without this 4:1 reduction.

The possibility that significant peaks might be lost during horizontal reduction was considered, and 5x1 quantile filters (with p = 0, 0.25, 0.75 and 1) were also tried for the initial filtering.

*vextzoom* was tried later as a substitute for *vsmzoom* in the last step.

## 5.2 Speckle removal

The functions *vspeckle, vquantile, vabseil, vicafil, vcntsmth2, vcntsmth3* and *vcleanpol* (see sections 3, 4) were tested with various parameter values on both full and horizontally-reduced sub-images.

For functions that assume a multiplicative-plus-additive noise model, there are parameters for both the noise components. For a log(power) image, it was expected that speckle would be approximately an additive term.

After some poor results, possible benefits of operating on grey-level transformed data were considered. The functions *vsetmap* and *vmapdata* were used to apply transformations directly to the 8-bit images. The combinations

square -> filter -> square-root

and

exp -> filter -> log

were tried, "filter" being one of the functions listed above. (Note that, for a grey level $0 \leq g \leq 255$, "square" here means $g^2/255$ and "exp" means $\exp\{g.\ln(256)/255\}-1$; "square-root" and "log" are the respective inverses. These definitions produce output suitable for 8-bit processing.)

The functions *vabseil, vcntsmth2, vcntsmth3* and *vcleanpol* also accept floating-point images. The log and exp transformations above can be applied to such images using standard Khoros functions. Floating-point calculations were tried to check for any problems with grey-level quantisation.

For *vabseil* and *vicafil* operating on exponentially transformed data, multiplicative noise was expected but low-level additive noise was also considered in setting parameters.

## 5.3 Quality of speckle removal output

Two tests have been used in MRD to measure the reduction of speckle and preservation of significant targets (N. Stacy, pers. comm.). These measure the amount of speckle remaining, and the degradation of bright single-pixel targets.

In the first test, a "feature-free" area of an image is filtered, and the coefficient of variation

$$V = \sigma/\mu$$

of the power is computed. This should be near 1.0 for unfiltered single-look SAR data, and as small as possible after filtering.

In the second test, an image with a single-pixel target is filtered and the ratio

$$S = (\text{log power of target pixel})/(\text{mean log power of 8 neighbouring pixels})$$

is computed from the output as a sharpness measure. This should be as large as possible.

Alternatives for the statistic in the second test were considered. Clearly some allowance should be made for power in the surrounding "background" pixels before processing, for this may be present in powers of both the target pixel and its neighbours after processing. If the target is sufficiently blurred that the statistic is near 1, the amount of blurring into more distant pixels should be taken into account. These requirements suggest a measure of spread like

$$\frac{\Sigma(\text{pixel power - background power})\text{x}(\text{pixel distance})^2}{\Sigma(\text{pixel power - background power})}$$

where the sums are taken over all pixels near the target. This measure can be related to a linear model in which power is diffused in eight directions, but for some kinds of blur (eg motion blur in visual images) some negative conductances are required in the model. If the model is valid, the spread is proportional to time; if two blurring operations are applied sequentially, the spread for their combination is the sum of their separate spreads.

It can further be argued that no combination of one target with one measure of spread is appropriate for all filters. This is because:

(i) Some filters (eg *vabseil*) are designed to blur small differences more readily than large ones. These will blur weak targets but leave strong ones unaffected;

(ii) Some filters (eg *vspeckle*) can create a weak halo around a sharp target, strong enough to be visually distracting when the target is very strong and speckle is weak;

(iii) Some filters (eg *vhmed*) can cause small targets to disappear or remain only as very weak and blurred patches; these actions are more serious problems than simple blur;

(iv) Some filters (eg *vcntsmth3*) reduce target sizes, causing small targets to disappear but allowing larger ones to remain.

The coefficient of variation V was computed after applying each filter to an apparently feature-free area, then transforming to relative power values. The sharpness S was computed for two single-pixel targets of grey level 100 and 200 respectively, on a speckle-free background of grey level 50, after applying each filter (but not transforming to power values). These targets are representative of targets on the sample data used for other tests, if the grey scale is taken to be logarithmic as in the sample data.

## 5.4 Thresholding

The *vthresel* function was tested on various sub-images as a general classifier, but particularly for its ability to separate local bright objects (presumed to be man-made, and sometimes identifiable as buildings etc). The effects of removing speckle first, grey-scale transformations (especially exponential), varying the desired number of classes, and using contour lengths rather than histograms, were tested in various combinations. (Zooming an image before thresholding should make little difference to the sub-ranges chosen.)

In choosing the number of classes, the criterion "residual < 0.05 x histogram area" was specified.

## 5.5 Subregion identification using local properties

The *vharalick* function was tested both for its ability to highlight man-made objects and for its ability to separate areas of saltbush and of scattered trees in training and test images. In these tests, because data was processed without removing speckle, a window with a 4:1 aspect ratio was used to process pixels corresponding to a square area on the ground. The function produces nine values at each pixel with wide dynamic ranges. Detection of man made objects was assessed by viewing the values of the texture features which had been logged and scaled. For the two class test, the average of the nine texture features over the training and test subimages was examined for its ability to differentiate between sample areas.

Since texture is inherently dependent on the scale of features of interest, a number of tests were made using different values of radius (separation between two pixels).

The function *vptron* was used to learn two classes (saltbush and scattered trees) and then classify each pixel into the most likely class. In practice, a classification such as this will involve many classes, but here two were used as an artificial example. The images used for training and testing were synthetically generated from real images by extracting known regions of trees and saltbush. The training and test (sub)images were taken from the same main image, but have no common pixels.

The results of the classification were compared with the "true" classes using the Kappa statistic of (Congalton, 1991). The effect of majority filtering through *vmajor* was also tested.

# 6. RESULTS

## 6.1 Geometric correction

The default order of smoothing was found most suitable for zooming the sample data with *vsmzoom*. Even with this order, ringing effects were very pronounced when speckle was still present in the image. After speckle removal, ringing was still noticeable near edges and point reflectors, but was no longer a real problem.

Using *vextzoom* with cubic convolution interpolation greatly reduced the ringing effects around speckle, edges and point reflectors.

Stretching the output of speckle-reducing filters in one direction always produced a "grainy" texture. The use of the 4:1 horizontal reduction approach generally produced a non-directional texture more pleasing to the eye. It had the further advantages of reducing data storage requirements and processing times by a factor of four.

The use of quantile filtering in the reduction step, instead of just median filtering, raised or lowered the mean output grey level but did not appear to give better preservation of point reflectors in the final result. The use of 5x1 mean filtering, or 3x1 median filtering, rather than 5x1 median filtering, did not appear to affect the results greatly.

## 6.2 Speckle reduction

Many combinations of geometric correction methods, filters and grey-scale transformations gave poor results. Use of the filters requires experimentation with parameters, and settings found to give good results for the sample data supplied may not be optimal for other data.

Histograms of small areas of the sample data showed that, in areas of low signal, the most common grey level was zero. The spread of the positive values in these areas suggested that negative log(power) values had occurred but had been clipped to zero during detection or processing. J. Douglass (pers. comm.) advises that the values have been "stretched"; this implies clipping at both ends of a selected range.

The following quality measures are given where available: the coefficient of variation V for filtered speckle, and the sharpness S for single-pixel targets of two intensities, as described in section 5.3 .

### 6.2.1 *vspeckle*

*vspeckle* gave poor results on the raw data, leaving dark spots in the image until enough iterations were applied to blur strong reflectors. (V=0.39, S=1.0, 2.0)

When applied to squared values, it reduced speckle to faint patchiness after 5 iterations, and preserved strong point reflectors and some edges well. (V=0.26, S=1.0, 2.0) Faint linear reflectors and weaker edges were blurred beyond 2 iterations.

On exponentially transformed data it gave good speckle reduction in a single iteration and preserved strong reflectors well. (V=0.30, S=1.1, 2.7)

4:1 data reduction did not improve the results of *vspeckle*. When combined with the exponential transformation it introduced artefacts into the results.

### 6.2.2 *vquantile*

*vquantile* gave poor results generally, and could not remove speckle without destroying or shrinking strong point reflectors. (V=0.73, single-pixel targets removed, for the median.) Transforming grey levels does not have any effect on this filter.

### 6.2.3 *vabseil*

*vabseil* improved the raw data after 25 iterations with σ(additive noise) = 5, or 5 iterations with σ(additive noise) = 15 (V=0.42, S=1.01, 2.5), but the residual speckle looked "grainy" after geometric correction.

For exponentially transformed data, the results were poor when pure multiplicative noise was assumed. Setting σ(additive noise) = 2 and σ(multiplicative noise) = 0.15 gave fair results in 5 iterations (V=0.45, S=1.0, 1.14). Figures 2-3 show the effect on a small area.

For 4:1 reduced, exponentially transformed data, good results were obtained from 3 iterations with σ(additive noise) = 1 to 1.5 and σ(multiplicative noise) = 0.15 to 0.2, but some finer features were lost (V=0.46). Working with floating-point images gave smoother results away from strong features. Untransformed data gave poor results.

### 6.2.4 *vicafil*

*vicafil* gave poor results until grey levels were transformed and both additive and exponential noise were assumed. For exponentially transformed data, results were good for σ(additive noise) = 2, σ(multiplicative noise) = 0.2 and a neighbourhood of radius 3 pixels (V=0.53, S=1.0, 4.0).

For 4:1 reduced data the results were similar.

### 6.2.5 *vcntsmth2* and *vcntsmth3*

*vcntsmth2* and *vcntsmth3* gave very poor results without 4:1 reduction (V=0.58 and V=0.72 respectively, single-point targets destroyed), possibly because of the correlation in the horizontal (along-track) direction.

After 4:1 reduction, *vcntsmth2* was eliminated because it produced distracting "chequerboard" artefacts (as it sometimes does with very noisy data). Using *vextzoom* rather than *vsmzoom* in the final zoom reduced the effect of these artefacts, but *vcntsmth3* avoided them altogether, and in two iterations produced apparently good results, removing fine details of speckle while preserving slightly larger features better than any other filter (V=0.49). Figures 2 and 4-6 show the effects of *vcntsmth3* on small and large scales.

Exponentially transforming the data made little difference when done with floating-point images, but degraded the results when done with 8-bit images.

These filters were not specifically designed for SAR application, and the apparent preservation of detail was surprising. Some of the finer output detail suggests weak, worm-like ground or vegetation structures. Such structures are not obvious in the raw images but are visible there on close inspection. Similar structures can be produced by filtering white Gaussian noise, so they probably do not give useful information about ground features.

Some narrow features (tracks?) were visible in the raw images but not in the filter output. These features are not easily distinguishable from speckle in, say, a 10x10 pixel window. Only when a much larger area is viewed and the feature can be followed over a long distance in a straight (or nearly straight) and possibly broken line does it become obvious to a human observer.

Similar (but more blurry) results were produced by applying a convolution filter, with kernel:

$$\begin{bmatrix} 1/16 & 2/16 & 1/16 \\ 2/16 & 4/16 & 2/16 \\ 1/16 & 2/16 & 1/16 \end{bmatrix}$$

to the 4:1 reduced image, using *vconvolvc*. This kernel was specifically chosen for attenuation of fine detail.

### 6.2.6 *vcleanpol*

*vcleanpol* gave mediocre results generally. The best compromise between removing coarse speckle patterns and keeping point reflectors sharp was found by stopping after about 16 iterations (V=0.54, S=1.0, 1.0).

## 6.3 Subregion identification

### 6.3.1 Thresholding

Thresholding did not produce any useful classifications without grey-level transformation. Both the square and exponential transformations designed for noise removal modified the histogram shape so that bright objects produced a long upper "tail" where a single threshold could be set.

Using the contour lengths rather than the histogram in *vthresel* gave much more reliable results.

When grey level transformation and the use of contour lengths were combined, reasonable classifications of "bright object" and "other" were obtained. The number of classes chosen was then usually two where bright objects were present and one (no threshold at all) otherwise. When more than two classes were allowed, very bright objects sometimes appeared as a third class. Speckle points were sometimes detected as small objects.

Removing speckle before thresholding did not greatly affect the sub-ranges chosen, but merely smoothed the outlines of the resulting objects. No further classification by grey level was possible.

### 6.3.2 Texture

Radius 1 measures appeared to give the most useful results from *vharalick*.

Compared to the raw image values, the first derived feature (mean) was more useful, but only because it reduced speckle, a task probably better performed by a specialized routine. Features 2 (variance), 4 (correlation) and 9 (cluster prominence) emphasised the sidelobes of the antenna pattern where there were bright spots in the image. Feature 8 (cluster shade) emphasised quickly varying sections of the image; these included man-made features such as roads, but also other apparently natural linear features. Features 3 and 5-7 were not considered useful.

A particular feature of the data brought out by texture processing was the effect of clipping of negative values in the darker areas (see section 6.2).

Features 2, 4, 7 and 9 had the power to clearly distinguish between the saltbush and tree classes, while features 1, 3 and 5 showed marginal discriminating power. Features 6 and 8 were found to provide misleading cues for class discrimination between land cover types.

### 6.3.3 Supervised classification

The synthetically generated images of saltbush and trees are presented in Figure 1 along with the resulting classification. Borders are left unclassified in order to make a clearer comparison with other classification systems and the settings of parameters in this system.

For an MLP with an input order = 9 (equivalent to finding texture in a 7x7 window) with 25 hidden nodes (structure = 49-25-2) a classification accuracy of 86.2% ( equivalent to Kappa = 0.72) was obtained. Kappa was increased to 0.87 after majority filtering the output with a radius of 3 and increased to 0.97 with a filter radius of 7. Realistically, this excellent value would not be attainable with small windows.

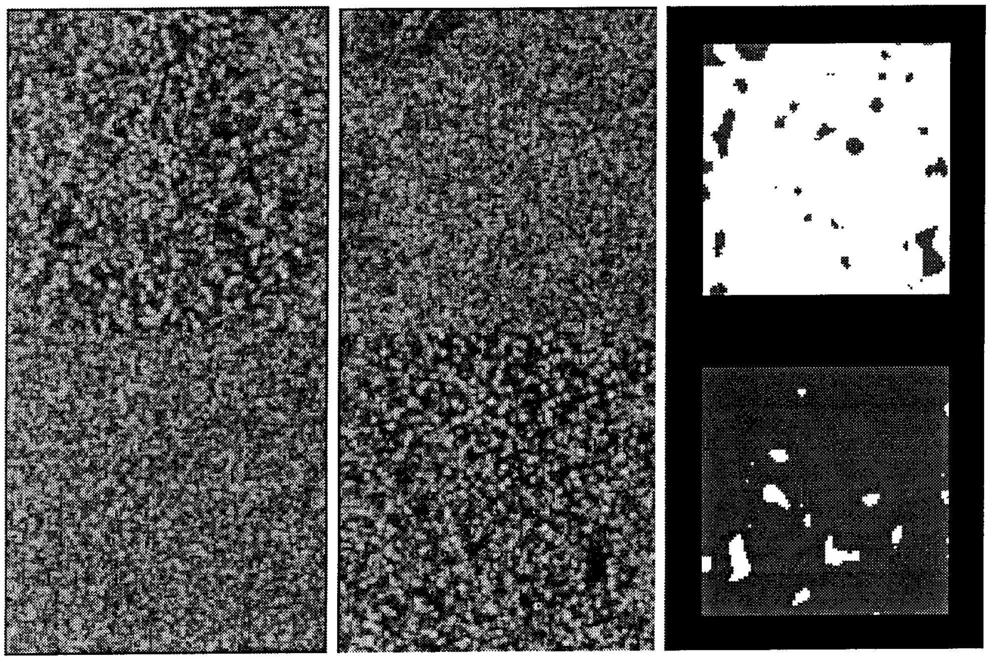


**Figure 1:** Training Image and Test Image

## 7. CONCLUSIONS

### 7.1 Geometric processing

Normal INGARA output includes 4-look processing, which makes pixels 4 times larger in the along-track direction and removes the geometric distortion, without the need for subsampling.

For the sample data used, 5x1 (along-track) median filtering, followed by extraction of every fourth column of the image, preserves all the important image information while giving a useful reduction of disk storage and processing time. This raises the interesting question of using different approaches for combining the looks in 4-look processing. Images produced by MRD's current 4-look processing are available but have not been investigated.

The functions available for the subsampling in Khoros are clumsy to use and *vextzoom* can do the same more conveniently (but less efficiently). A specialised function with the filtering built in would be better.

Subsampling also affects the speckle-reduction process, improving it in the results from some filters for the sample data supplied.

The *vsmzoom* and *vextzoom* functions are useful for displaying the speckle-reduced image at higher resolutions. It seems fairly clear, however, that the 4x1 pixels in an Ingara image before 4-look processing do not contain sufficient information to be able to reconstruct an accurate high resolution 4x4 pixel view of the underlying data. Rather the best that can be done is to perhaps improve on the low resolution 1x1 view given by 4-look processing by using filters tuned to the particular characteristics of speckle noise. If the image must be viewed at higher magnification then it can be zoomed by either *vsmzoom* and *vextzoom*, but little will be gained by zooming it by more than a factor of two.

There are two reasons for this. The first is the presence of speckle; it obviously degrades the fine scale detail in the 4x1 pixels view that is essential for a successful high resolution 4x4 reconstruction. The second, and more important, is that there is simply not enough information in 4x1 pixels to reliably reconstruct the missing 12 pixels in a high resolution 4x4 view. Dr. Chris Woodruff in LSOD has explored the ability of zooming algorithms to improve human perception of details in imagery: in summary, his findings are that zooming by a factor of two gives up to ~20% improvement in target detection, but that further zooming did not lead to further detection (Woodruff and Newsam, 1994).

## 7.2 Speckle removal

The method chosen for speckle removal will depend on what other image features are most important to preserve.

For the most complete elimination of fine speckle, with fairly good retention of all coarser features, the *vcntsmth3* function, with 4:1 reduction but without grey-level transformation, is most useful. This is the recommended approach when large areas are viewed with further subsampling (see Figures 5 and 6).

For preservation of strong point and line reflectors, the standard *vspeckle* function, or the *vabseil* function (operating on exponentially transformed data with appropriate settings for both additive and multiplicative noise) are useful. *vsetmap* provides a simple way to perform the grey-level transformations when the filter depends on 8-bit data, but otherwise floating-point processing gives better results (see Figures 2-4).

The ability to preserve very narrow line features may depend on using filters with very large window sizes and testing specifically for persistent lines.

## 7.3 Thresholding

The *vthresel* function, when used with its "contour length" option on data with suitably transformed grey-levels (squared or exponentiated), can detect small bright objects. Speckle need not be removed first. The function is not forced to find objects in every image, and may be set to distinguish exceptionally strong objects by allowing more than two classes in the output.

When objects are detected, some speckle points may appear with them. Further processing to remove very small objects may then be needed.

## 7.4 Subregion identification using local properties

When the distributions of grey levels from different types of target overlap significantly, texture is the only feature available for subregion identification.

Texture in the sample data was affected by the clipping of negative log(power) values to zero. This was more common in areas where the mean cross-section was low. Better results might be obtained if full floating-point data were used.

Two different kinds of texture identification were tested: pattern recognition - straight identification of regions presented one at a time; and segmentation - identifications of regions in the context of other regions. On the given data the texture features generated by *vharalick* were not useful. However, the function *vptron* was able to learn textures and classify textures accurately (better than 90%). This technique could be useful for groundcover or surface geology identification.

## REFERENCES

| | |
|---|---|
| Alvarez L., Lions P. & Morel J. | "Image selective smoothing and edge detection by non-linear diffusion II", SIAM J. Numer. Anal., 29(3), 1992, pp 845-866. |
| Congalton R.G. | "A review of assessing the accuracy of classifications of remotely sensed data", Remote Sensing of the the Environment, 37, 1991, pp 35–46. |
| Crimmins T.R. | "Geometric filter for reducing speckle", Optical Engineering, 25, 1986, pp 651-654. |
| Haralick R.M., Shanmugan K. & Dinstein I. | "Textural features for image classification", IEEE Trans. Systems, Man and Cybernetics, 3(6), 1973, pp 610--621. |
| Konstantinides K. & Rasure J.R. | "The Khoros software development environment for image and signal processing", IEEE Trans. Image Processing, 3(3), 1994, pp 243-252. |
| Laws K.I. | "Textured image segmentation", Department of Electrical Engineering, Image Processing Institute, University of Southern California, Los Angeles, California 90007, 1980. |
| Lee J. | "A simple speckle smoothing algorithm for synthetic aperture radar images", IEEE Trans. Systems, Man Cybernetics, 13, 1983, pp 85-89. |
| McDonald D.M. | "Speckle reduction in synthetic aperture radar images", Technical Memorandum SRL-0010-TM, DSTO Surveillance Research Laboratory, 1988. |

| | |
|---|---|
| Newsam G.N. | "Smooth zooming of images", Proc. First New Zealand Conf. on Image & Vision Computing, 1993, pp 221-228. |
| Rasure J.R., Williams C., Argiro D. & Sauer. | "A visual language and software development environment for image processing", International J. Imaging Systems and Technology, 2, 1990, pp 183-199. |
| Stacy N.J.S. & Burgess M.P. | "The Australian airborne radar surveillance system", Proc. 7th Australasian Remote Sensing Conf., 1994, pp 205-212. |
| Unser M. | "Sum and difference histograms for texture classification", IEEE Trans. Pattern Analysis and Machine Intelligence, 8(1), 1986, pp 119-125. |
| Whatmough R.J. | "Two new non-linear filters for removing strong noise from sharp images", Proc. Fifth Australasian Remote Sensing Conf., Perth, 1990, pp 1283-1290. |
| Whatmough R.J. | "Automatic threshold selection from a histogram using the "Exponential Hull"", CVGIP: Graphical Models and Image Processing, 53(6), 1991, pp 592-600. |
| Whitbread P.J. | "Multi-spectral Texture: Improving Classification of Multi-spectral Images by the Integration of Spatial Information", PhD thesis, University of Adelaide, 1992 |
| Woodruff C.J. & Newsam G.N. | "Displaying undersampled imagery, Optical Engineering", 33, 1994, pp 579-585. |

Figure 2. Sample area 1 (512x128 pixels, zoomed x4 vertically)

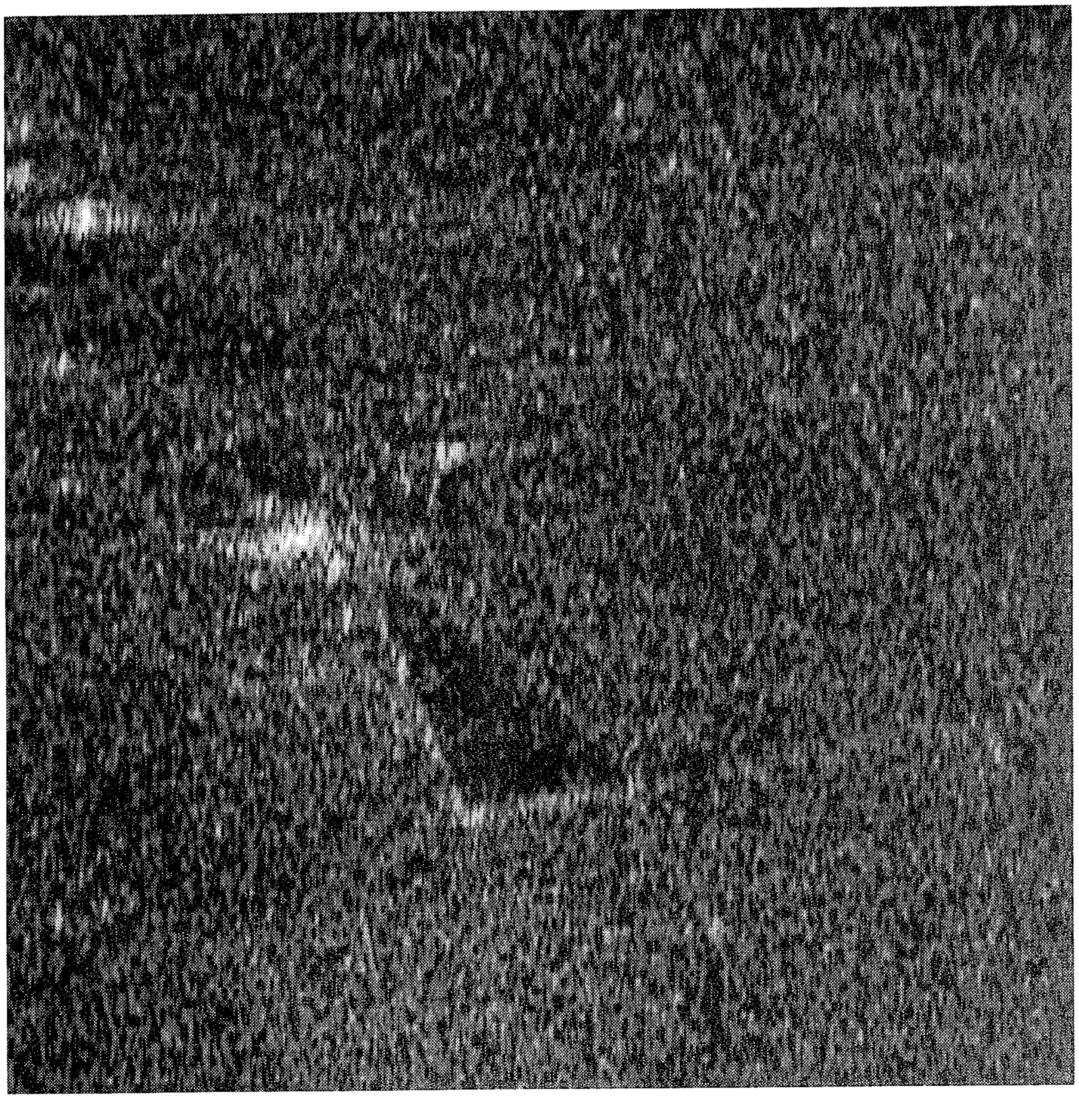


Figure 3. Sample area 1 (exponentially transformed,filtered by *vabseil* , then zoomed x4 vertically)

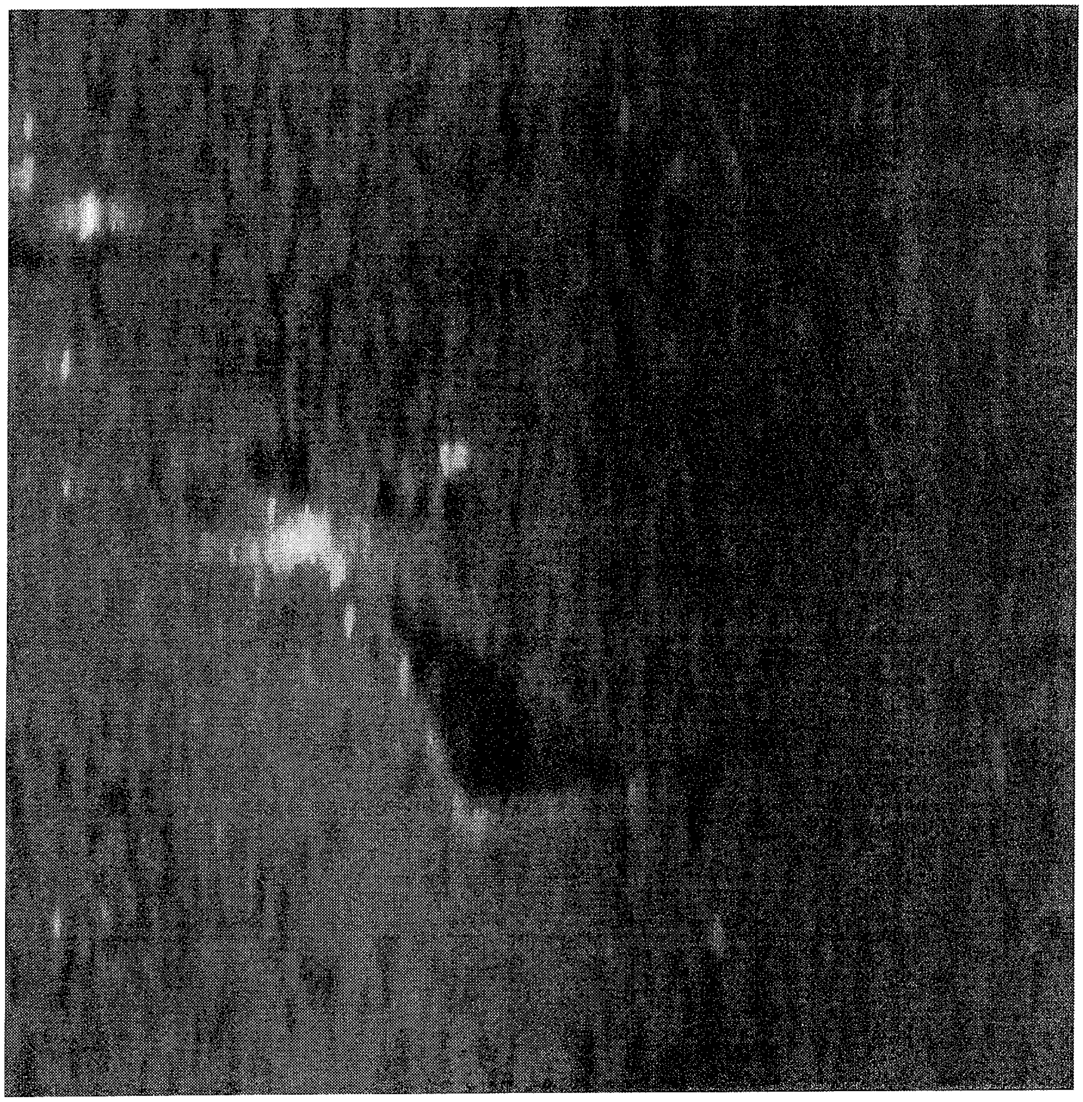


Figure 4. Sample area 1 (subsampled x4 horizontally, exponentially transformed, filtered by *vcntsmth3* then zoomed x4 both ways)

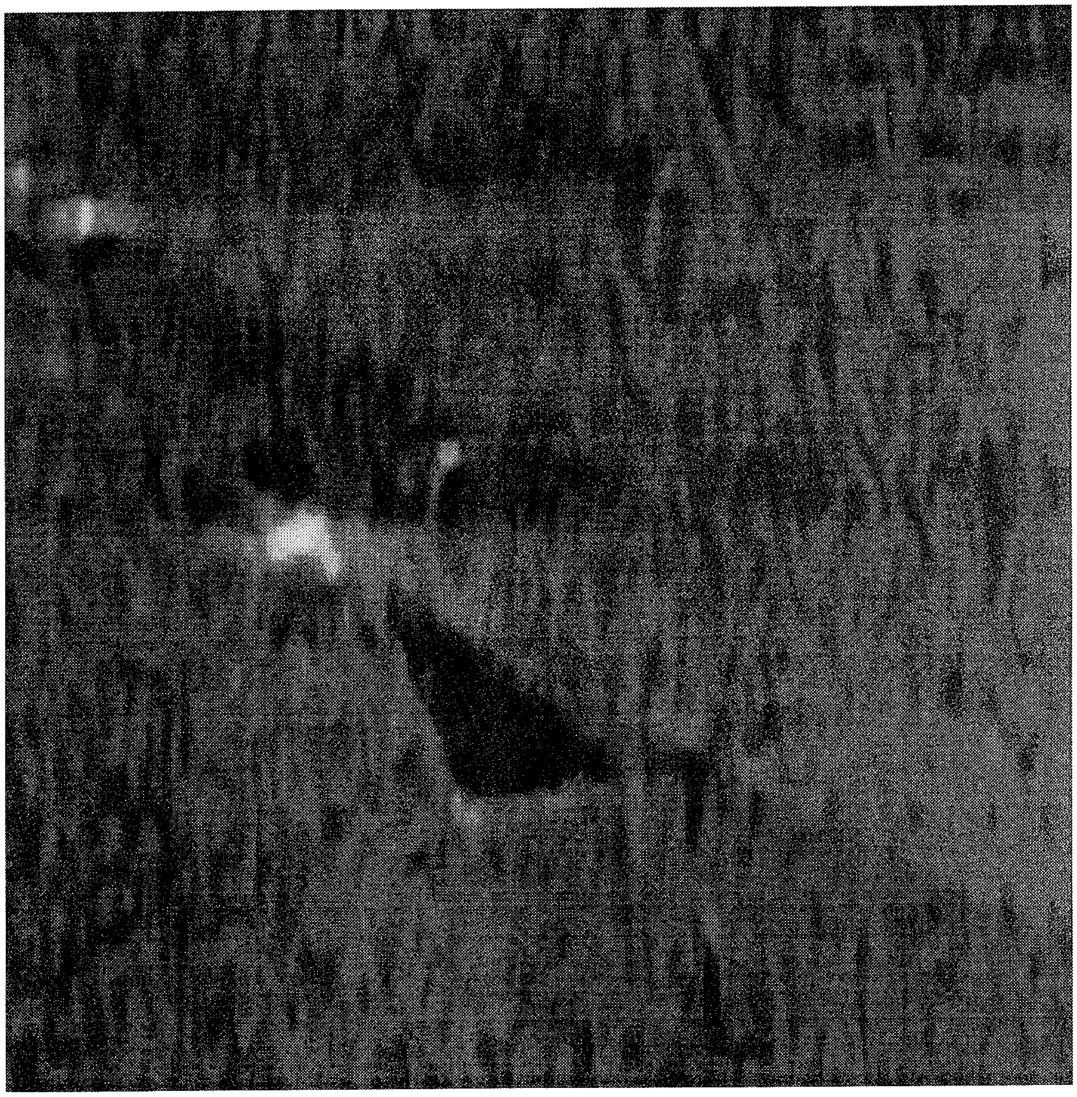


Figure 5. Sample area 2 (2000x500 pixels, subsampled x4 horizontally)

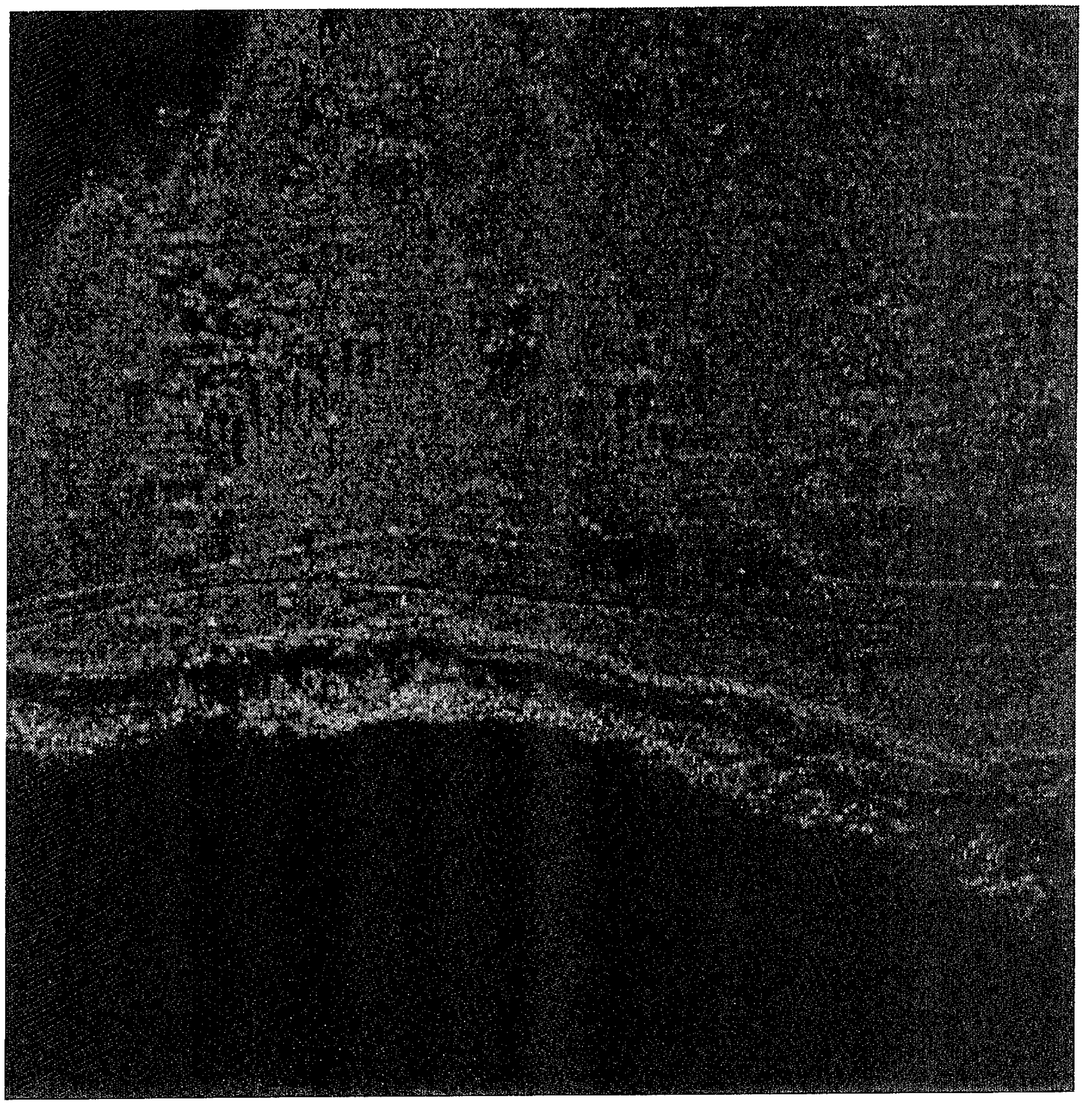
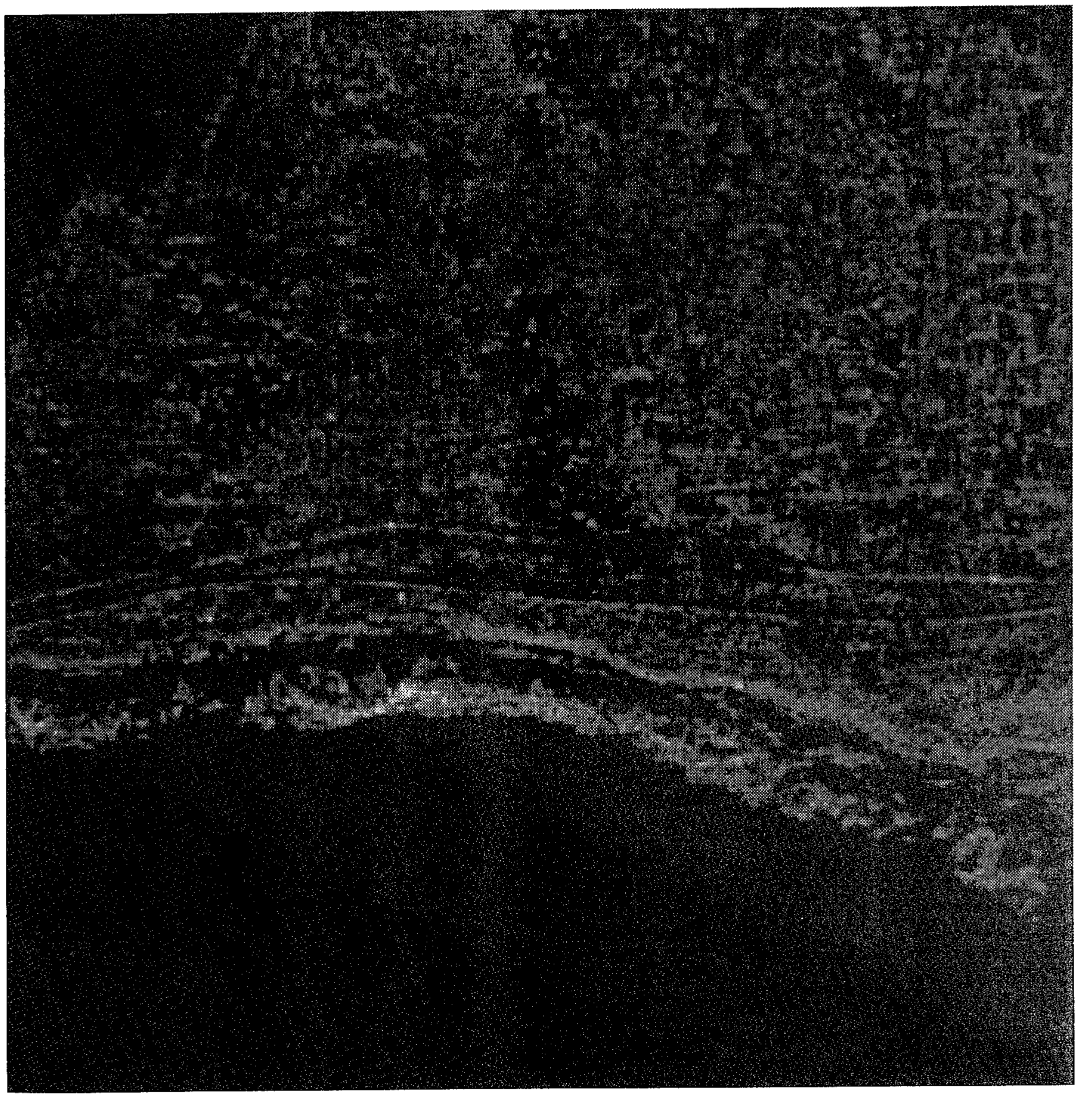


Figure 6. Sample area 2 (subsampled x4 horizontally, exponentially transformed, then filtered by *vcntsmth3*)



## DISTRIBUTION

| | No. of Copies |
|---|---|
| **Defence Science and Technology Organisation** | |
| Chief Defence Scientist ) | 1 shared copy |
| Central Office Executive ) | |
| Counsellor, Defence Science, London | Cont Sht |
| Counsellor, Defence Science, Washington | Cont Sht |
| Senior Defence Scientific Adviser | 1 copy |
| Scientific Adviser POLCOM | 1 copy |
| Director, Aeronautical & Maritime Research Laboratory | 1 copy |
| Chief Maritime Operations Division | Cont Sht |
| Chief Air Operations Division | Cont Sht |
| Chief WeaponsSystems Division | Cont Sht |
| **Navy Office** | |
| Naval Scientific Adviser | 1 copy |
| **Army Office** | |
| Scientific Adviser, Army | 1 copy |
| **Airforce Office** | |
| Air Force Scientific Adviser | 1 copy |
| **Defence Intelligence Organisation** | |
| Assistant Secretary Scientific Analysis | 1 copy |
| **Electronics & Surveillance Research Laboratory** | |
| Director | 1 copy |
| Chief Information Technology Division | 1 copy |
| Chief Electronic Warfare Division | Cont Sht |
| Chief Communications Division | Cont Sht |
| Chief Land, Space and Optoelectronics Division | Cont Sht |
| Chief High Frequency Radar Division | Cont Sht |
| Chief Microwave Radar Division | Cont Sht |
| Research Leader Command & Control and Intelligence Systems | 1 copy |
| Research Leader Military Computing Systems | 1 copy |
| Manager Human Computer Interaction Laboratory | Cont Sht |
| Executive Officer, Information Technology Division | Cont Sht |
| Head, Trusted Computer Systems Group | Cont Sht |
| Head, Command Support Systems Group | 1 copy |
| Head, Exercise Analysis Group | Cont Sht |
| Head, C3I Systems Engineering Group | Cont Sht |
| Head, Computer Systems Architecture Group | Cont Sht |

## DISTRIBUTION (CONT)

| | No. of Copies |
|---|---|
| Head, Systems Simulation and Assessment Group | 1 copy |
| Head, Information Management Group | 1 copy |
| Head, Intelligence Systems Group | 1 copy |
| Head, Information Acquisition & Processing Group | 1 copy |
| Head, Radar Techniques and Signal Processing, MRD | 1 copy |
| Head, Radar Systems, MRD | 1 copy |
| Head, Target Identification, MRD | 1 copy |
| Head, Image Processing, LSOD | 1 copy |
| Dr G. Newsam | 1 copy |
| Dr N. Redding | 1 copy |
| Mr J. Douglass | 1 copy |
| Dr N. Stacy | 1 copy |
| Mr W. Roberts | 1 copy |
| Mr R. Whatmough (Author) | 2 copies |
| Mr P. Whitbread (Author) | 2 copies |
| Publications & Publicity Officer ITD | 1 copy |
| **Libraries and Information Services** | |
| Australian Government Publishing Service | 1 copy |
| Defence Central Library, Technical Reports Centre | 1 copy |
| Manager, Document Exchange Centre, (for retention) | 1 copy |
| National Technical Information Service, United States | 2 copies |
| Defence Research Information Centre, United Kingdom | 2 copies |
| Director Scientific Information Services, Canada | 1 copy |
| Ministry of Defence, New Zealand | 1 copy |
| National Library of Australia | 1 copy |
| Defence Science and Technology Organisation Salisbury, Research Library | 2 copies |
| Library Defence Signals Directorate Canberra | 1 copy |
| British Library Document Supply Centre | 1 copy |
| Parliamentary Library of South Australia | 1 copy |
| The State Library of South Australia | 1 copy |
| **Spares** | |
| Defence Science and Technology Organisation Salisbury, Research Library | 6 copies |

Department of Defence

**DOCUMENT CONTROL DATA SHEET**

| | | | |
|---|---|---|---|
| | | | 1. Page Classification Unclassified |
| | | | 2. Privacy Marking/Caveat ( of document ) |
| 3a. AR Number AR-008-799 | 3b. Laboratory Number DSTO-TR-0146 | 3c. Type of Report Technical Report | 4. Task Number 93/163 |
| 5. Document Date JANUARY 1995 | 6. Cost Code 846387 | 7. Security Classification *U Document, U Title, U Abstract | 8. No. of Pages 44 |
| | | | 9. No. of Refs. 16 |

7. S (Secret) C (Confi ) R (Rest) U (Unclass)

* For UNCLASSIFIED docs with a secondary distribution LIMITATION, use (L) in document box.

10. Title

FUNCTIONS FOR PROCESSING SYNTHETIC APERTURE RADAR IMAGERY

11. Author(s)

Robert Whatmough and Paul Whitbread

12. Downgrading/Delimiting Instructions

N/A

13a. Corporate Author and Address

Electronics & Surveillance Research Laboratory
PO Box 1500, Salisbury SA 5108

13b. Task Sponsor

DEF

14. Officer/Position responsible for

Security:.....SOESRL........................................................

Downgrading:.....CITD............................................

Approval forRelease:...CITD............................................

15. Secondary Release Statement of this Document

APPROVED FOR PUBLIC RELEASE

16a. Deliberate Announcement

No Limitation

16b. Casual Announcement (for citation in other documents)

[✓ No Limitation] ☐ Ref. by Author , Doc No. and date only.

17. DEFTEST Descriptors

Synthetic aperture radar
Image enhancement
Ingara
Khoros

18. DISCAT Subject Codes

N/A

19. Abstract

The INGARA synthetic aperture radar system can potentially benefit from higher level algorithms and software to enhance its basic imagery so that important features are obvious to human operators. These could include filters for removing speckle, zooming functions for correcting the aspect ratio of the imagery and classifiers for distinguishing ground cover types. The system could also make use of a development environment in which such functions can be easily developed and trialled. This Note describes the Khoros image processing system and functions added to that system by researchers in ITD that could be of use in processing INGARA data. It also includes results of tests of the effectiveness of these functions in enhancing sample INGARA data.